From Vocal Instructions to Household Tasks: The Inria Tiago++ in the euROBIN Service Robots Coopetition

Fabio Amadio, Clemente Donoso Krauss, Dionis Totsila, Raphael Lorenzo, Quentin Rouxel, Olivier Rochel, Enrico Mingo Hoffman, Jean-Baptiste Mouret, Serena Ivaldi

▶ **To cite this version:**

Fabio Amadio, Clemente Donoso Krauss, Dionis Totsila, Raphael Lorenzo, Quentin Rouxel, et al.. From Vocal Instructions to Household Tasks: The Inria Tiago++ in the euROBIN Service Robots Coopetition. 2024. hal-04868440

HAL Id: hal-04868440
https://hal.science/hal-04868440v1

Preprint submitted on 6 Jan 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

From Vocal Instructions to Household Tasks: The Inria Tiago++ in the euROBIN Service Robots Coopetition

Fabio Amadio*, Clemente Donoso*, Dionis Totsila*,
Raphael Lorenzo, Quentin Rouxel, Olivier Rochel, Enrico Mingo Hoffman, Jean-Baptiste Mouret, Serena Ivaldi

***Abstract*—This paper describes the Inria team's integrated robotics system used in the 1st euROBIN *coopetition*, during which service robots performed voice-activated household tasks in a kitchen setting. The team developed a modified Tiago++ platform that leverages a whole-body control stack for autonomous and teleoperated modes, and an LLM-based pipeline for instruction understanding and task planning. The key contributions (opens-sourced) are the integration of these components and the design of custom teleoperation devices, addressing practical challenges in the deployment of service robots.**

***Index Terms*—AI-Enabled Robotics; Domestic Robotics; Telerobotics and Teleoperation; Software-Hardware Integration for Robot Systems**

I. Introduction

This paper describes the system integration and the software/hardware modules used by the Inria team participating to the 1st euROBIN *coopetition* (i.e., cooperative competition, where teams are rewarded when they collaborate in sharing software), which took place in Nancy, France, on the 25-28 November 2024. EuROBIN is a Network of Excellence in AI and Robotics, funded by the European Commission. Among its objectives, it promotes transfer of robotics and AI software, methods and practices, by organizing annual robotics events where several teams collaborate to solve challenging tasks. Twenty teams participated to the 1st *coopetition*, in three different leagues. The Inria team participated to the *Service Robots League*, including six teams, where mobile manipulators interact with people and objects in a domestic environment, similar to the *RoboCup@Home* [1].

Service or domestic robots must possess navigation and manipulation skills, as well as the ability to understand and interpret commands from humans, and act accordingly [2]. Localization, perception and control are critical to navigate in a cluttered environment and interact with objects: doing this robustly in environments different from the lab is still challenging. For interaction, Large Language Models (LLMs) recently showed great potential in connecting natural language to robotic actions [3], [4], relying on "common sense" reasoning to comprehend ambiguous instructions; but they must be both reliable and fast enough for real-time human-robot interaction.

To help teams address these problems, the euROBIN *coopetition* introduced a simplified kitchen scenario in which the robot was requested to understand and execute standard instructions following two patterns: (1) *pick an object from a designated location and place it at another location*; (2) *pick an object from a designated location and deliver it to a person*. Teleoperation was allowed (but penalized in the point system) to extend the use of the platforms to new or unexpected situations and cope with failures that might occur during autonomous operation. The *coopetition* participants developed both hardware and software, addressing the challenges related to the system integration, including Third-Party software integration, and execution in realistic competition setting.

This work was supported by the the EU Horizon project euROBIN (GA n.101070596), the France 2030 program through the PEPR O2R projects AS3 and PI3 (ANR-22-EXOD-007, ANR-22-EXOD-004), the Agence Innovation Defense (ATOR project), the CPER CyberEntreprises, and the Creativ'Lab platform of Inria/LORIA.
All the authors are with INRIA, Université de Lorraine, CNRS, 54000 Nancy, France. (*) Equal contribution.

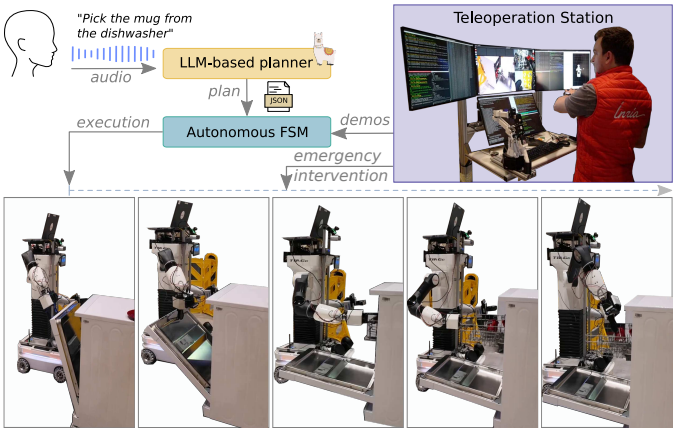

Fig. 1. System overview: our LLM-based planner understands voice commands and generate a plan (in JSON format) that is used to assemble a Finite State Machine (FSM) for carrying out the instruction. Teleoperation is used both to record expert demonstrations, and to intervene in case of emergency or failure.

Inria's robot (Fig. 1) is a modified Tiago++ with omnidirectional base. Its main components are a Whole-Body Control (WBC) stack for both teleoperated and autonomous operation and a LLM-based plan generation pipeline for instruction understanding. Together with the description of the system components and their integration (Fig. 2), we share the software and the design of the bimanual teleoperation devices (links reported in Table I). A video of the robot in action is available at *https://youtu.be/5mSIYuH4Mdk*.

II. Modules and Components

A. Motion control

This module coordinately control all the Tiago++ DOFs through an optimization-based WBC [5] stack that reads the proprioceptive data from the robot and maps user-level Cartesian references into joint-space commands for the low-level controller. This WBC formulation offers a principled

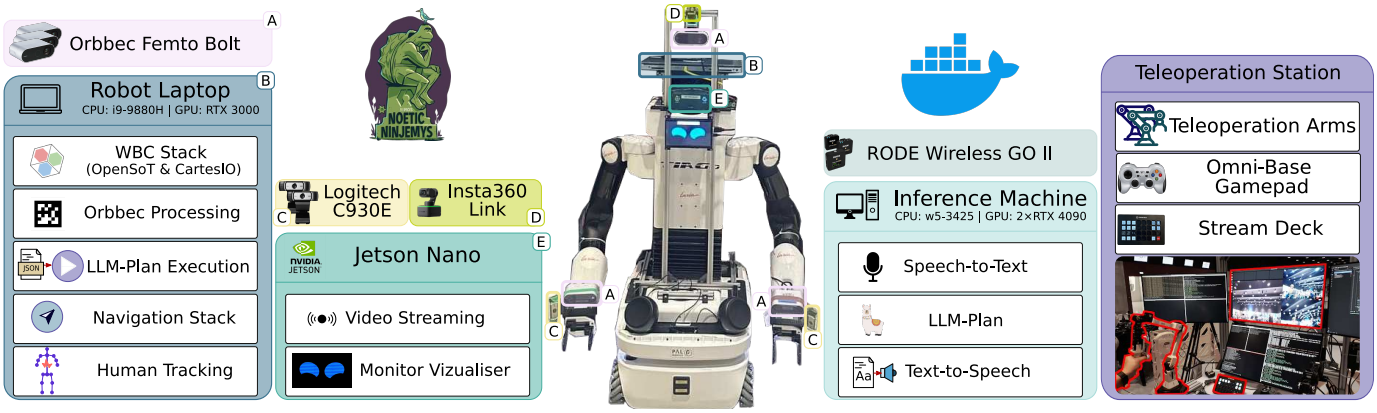

Fig. 2. General overview of our system, based on a dual-arm Tiago++ robot with an omnidirectional mobile base. Each block includes its connected peripherals, positioned above it. The robot is equipped with three RGB-D cameras (one fixed and two mounted on the grippers) and three webcams for teleoperation. A laptop mounted on the robot manages the WBC stack (Sec. II-A), RGB-D camera processing (Sec. II-C), plan execution (Sec. III), navigation (Sec. II-F) and human pose tracking (Sec. II-E). Additionally, a Jetson Nano on the robot streams webcam footage to the teleoperation station and renders interactive visuals on a 7-inch screen mounted on the robot. Computationally intensive tasks, such as LLM-based plan generation (Sec. II-G), are offloaded to a remote machine equipped with two GPUs. Finally, the Teleoperation Station (Sec. II-B) serves as a platform for (a) collecting demonstrations and (b) controlling the robot during failures or emergencies. We employed ROS Noetic middleware to integrate the different units (all running in dedicated Docker containers), connecting them to the Tiago++'s internal PC (where the `roscore` is running).

HW/SW Module	*Link*
OpenSoT	*github.com/ADVRHumanoids/OpenSoT*
CartesI/O	*github.com/ADVRHumanoids/CartesianInterface*
config	*github.com/hucebot/tiago_dual_cartesio_config*
Teleoperation	*github.com/hucebot/dxl_6d_input/tree/bimanual-teleoperation*
StreamDeck	*github.com/hucebot/stream_deck_controller*
AprilTags generator	*github.com/hucebot/april_tag_generator*
AprilTags detector	*github.com/hucebot/orbbec_apriltag_ros*
Human Tracking	*github.com/hucebot/eurobin_human_tracking*
LLM-based planner	*github.com/hucebot/eurobin_llm_plan*
Faster-Whisper*	*github.com/SYSTRAN/faster-whisper*
llama-cpp*	*github.com/ggerganov/llama.cpp*
fsm_cartesio	*github.com/hucebot/fsm_cartesio*
smach*	*wiki.ros.org/smach*

* Third-Party software

TABLE I
LINKS TO THE USED HARDWARE AND SOFTWARE COMPONENTS.

and unified solution to handle the redundancy of the platform (e.g., combining arm motions and torso motions), while taking into account joint position and velocity limits and avoiding self-collisions. This WBC stack is based on the CartesI/O [6] framework and the OpenSoT [7] library, which enables instantaneous control by formulating and solving Quadratic Programming (QP) problems using atomic entities like *Tasks* and *Constraints*. CartesI/O automates the setup of OpenSoT problems via configuration files and provides interfaces for interacting with tasks and constraints, offering APIs in C++ and Python, and supporting frameworks like ROS.

The control problem consists of three Cartesian tasks, formulated at the velocity level and arranged in a soft priority hierarchy: the left and right arm tasks, and the control of the omni-directional base. In the null-space, a postural task is employed to stabilize the robot's self-motions around the desired nominal configuration. The control of the base is achieved by modeling the robot as a floating-base system, constraining the omni-directional motion only on the ground.

We employ an open loop control scheme, sketched in Fig. 3, initialized at the initial references in generalized coordinates $\left[\mathbf{Q}_{r,0}^T,\, \mathbf{q}_{r,0}^T\right] \in SE(3)+\mathbb{R}^{19}$ retrieved when the controlled is started. At every control loop, the model is updated with the integrated output of the QP, namely $\left[\boldsymbol{\nu}^T\ \dot{\mathbf{q}}_d^T\right] \in \mathbb{R}^{6+19}$, as well as the postural task at the secondary priority level. The Cartesian reference commands are defined using homogeneous poses, $\mathbf{T}_r \in \mathbb{R}^{4\times4}$. A properly computed error, specifically the quaternion error, is used for the orientation, is calculated with respect to the forward kinematics. This error is then multiplied by a positive definite gain matrix, $\mathbf{K}_C \in \mathbb{R}^{6\times6}$, and combined with feed-forward Cartesian velocities, $\mathbf{v}_r \in \mathbb{R}^3$ for linear motion and $\boldsymbol{w}_r \in \mathbb{R}^3$ for angular motion. Similar considerations are done for the lower priority postural task, where the desired posture $\mathbf{q}_d \in \mathbb{R}^{15}$ can be adjusted during task execution.

The WBC scheme is implemented as a ROS node running at 250 Hz using the CartesI/O API. The output of this node is retrieved by another node, namely `ros_control_bridge`, in charge to dispatch the solution, meaning sending joint positions commands through a ROS topic interface as a `JointTrajectory` message to each kinematic chain controller, which is part of the `ros_control` layer of Tiago++. Meanwhile, base velocities are sent as `Twist` messages to the `cmd_vel` topic of the base controller.

While this solution may have some troubles in real-time settings and concerning synchronization between upper-body and base motions, we found it suitable for the tasks the robot has to perform at moderate velocity.

B. Teleoperation

The teleoperation interface is used both to record demonstrations (Sec. II-D) and as as fallback mode when the autonomous mode fails. It is designed around two low-cost master arms (Dynamixel actuators, used passively as encoders) inspired by the Aloha project [8], whose end-effector position (forward kinematics) is computed with the Pinocchio library [9]. Each of them consists of

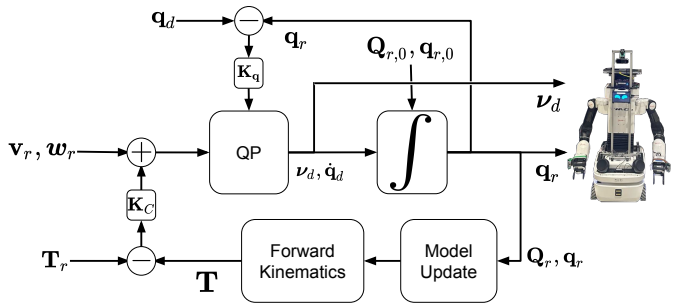

Fig. 3. Scheme of our QP-based WBC (cf. Sec. II-A).

seven motors: six control the full pose of the end-effector, while the seventh commands the opening and closing of the gripper. This configuration effectively maps the six DOFs required to perform all necessary motions despite the DOFs asymmetry [10]. However, it introduces mismatches between joint limits of the teleoperation device and the robot and can reduce the workspace compared to that of the robot.

A gamepad is used to send angular and linear velocity commands to the mobile base. All the commands are send to CartesI/O through its Python ROS client at 100 Hz.

Three cameras are used: one mounted on each end-effector of the robot (Logitech C930E and a 3-DOF controllable orientation camera Insta 360 Link, positioned as the "head" of the robot. The video is streamed on UDP using a Gstreamer pipeline GStreamer that leverages the on-camera hardware H.264 encoder, which keeps the latency of the stream low. Finally, a Stream Deck XL is used is to activate standard routines (e.g., homing) with the press of hardware buttons.

C. Object pose estimation

The robot needs to estimate the 6D poses of objects and key elements in the environment from RGB-D images for manipulation and navigation. We relied on AprilTag [11] fiducial markers. Although this approach is a substantial simplification of the problem, it is a reliable pose estimation module to integrate in our system, that will be eventually easily replaced by a 6D-pose estimation module in the future (e.g. [12]).

Our system relies on two RGB-D cameras (Orbecc Femto Bolt), different from the cameras used for teleoperation: one on the left wrist (for low-range detection), and one on top of the torso (for long-range detection). The tag IDs and the pixel coordinates of the four corners are extracted from the camera's color images using the AprilTag library [13]. Contrary to classic tag-based tracking, the tags are identified on the RGB image, but the 3D position and orientation of each marker in the camera frame are computed using the ordered point cloud from the camera's depth (time-of-flight) sensor. Each marker pose is then calculated and broadcasted on the ROS `tf` tree, which makes it possible to combine it with the forward kinematics of the robot to express the position in the base frame. In our experiments, this RGB-D approach provided more accurate and less noisy estimations, particularly for orientation and distance, than relying solely on AprilTag processing of color images.

D. Teaching object-centric manipulation skills

Many of the considered manipulation tasks (e.g., opening the dishwasher or the cabinet) require the execution of complex end-effector trajectories that are hard to program explicitly, but that can be more easily executed by re-playing demonstrations. First, the end-effector trajectories, gripper commands, and pose of the target object's AprilTag, were recorded during teleoperated task execution. In post-processing, we express the demonstration in the reference frame of the object, similarly to [14], obtaining a object-centric manipulation trajectory. During autonomous execution, the trajectory is loaded and transformed back in the robot base link by considering the current pose of the object tag. This approach proved sufficiently robust to cope with small deviations w.r.t. the original scenario (e.g., different orientation or position of the object to pick) while still relying on demonstrations. This system could be replaced with an image-based policy learning system [8]; this would remove the need for an object tracker and could generalize across more diverse scenarios, but would require many more demonstrations (in our system, a single high-quality demonstration is used for each task).

E. Human tracking

To autonomously carry out the handover instruction, we detect and track the person in the kitchen from the RGB-D camera stream (torso camera). Our approach involves five steps: (1) detecting humans, (2) tracking them, (3) estimating their 2D position in the RGB image, (4) computing the 3D coordinates using depth, and (5) determining if the persons are attentive (i.e., are looking at the robot) by checking their head pose. The closest attentive individual is selected as the handover target and their pose is broadcasted on the ROS `tf` tree.

Human detection and pose estimation were performed using YOLOv3 [15] for bounding box detection and ViTPose [16] for human pose estimation, implemented through the MMDetection [17] and MMPose [18] libraries. For tracking, we utilized the IoU-based tracker from MMPose with a threshold of 0.3, running at 10 fps. We filtered detections based on bounding box size to mitigate latency issues in pose estimation.

We used 6DRepNet [19] to estimate the head's 6D pose as a reliable proxy for gaze direction, given its robustness under challenging conditions compared to gaze estimation methods that depend on high-resolution images and good lighting around the eyes. Since the model is designed for front-facing heads, we applied it only when a rough heuristic indicated that the person was facing the camera. This was determined by the pixel difference between the detected left-ear and right-ear keypoints (greater than a threshold of 30 pixels).

F. Navigation

We relied on a basic navigation node that implements a ROS `MoveBaseAction` action to move the base at the desired offset w.r.t. a reference frame (e.g., AprilTag, tracked human, robot's base link) present in the `tf` tree. Together with simple, ad-hoc obstacle detection using the onboard LiDARs, and target searching procedures, this solution proved sufficient

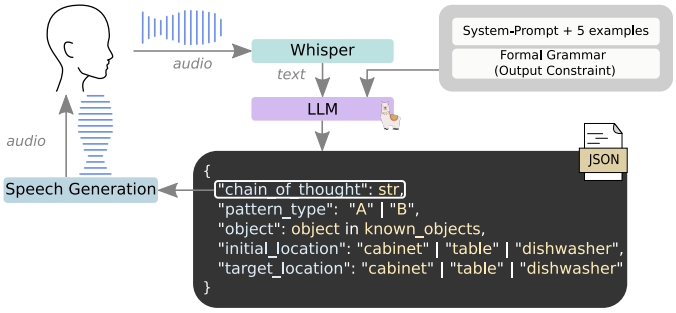

Fig. 4. LLM-based plan generation overview (cf. Sec. II-G).

in the context of the *coopetition* and eliminates the need to build a map of the environment. In future developments, it would be possible replace this simple solution with state-of-the-art navigation algorithms [20] without the need of modifying the overall system architecture.

G. LLM-based plan generation

In the euROBIN *coopetition*, the robot must be able to understand the verbal instruction that describes the assigned task. While structured human commands can be handled using traditional approaches like SnipsNLU [21], the use of LLMs offers a significant advantage in processing more general, unstructured inputs expressed by a human in natural language. LLMs can infer well-structured outputs even from ambiguous or conversational commands, deal with synonyms or paraphrases, and their capability to generate extra textual fields, such as chain-of-thought reasoning [22] improves interpretability.

We integrate Faster-Whisper, an optimized variant of OpenAI's Whisper [23], for speech-to-text processing. The transcribed commands are then passed to a pre-trained LLM (Llama-3.1-8B-Instruct) [24], along with a task-specific system prompt, which provides a detailed explanation of the expected output format, define the LLM's task, and request to generate a chain-of-thought reasoning before reaching any conclusions. The chain-of-thought reasoning generated by the LLM serves a dual purpose: (1) *improved output quality*, by ensuring that the plan adheres to task-specific constraints, providing logical, step-by-step clarity for the robot's actions; (2) *enhanced user communication*, we use Coqui TTS [25] to verbally communicate the chain-of-thought to the user. To further guide the LLM, we include five example input-output pairs, a technique known as Few-shot prompting [26], which has proven effective in LLM-based tasks [27]. Additionally, by using llama-cpp, we constrain the LLM output through grammar-based token sampling and acceptance [28], ensuring consistently parsable JSON output describing the plan (Fig. 4).

III. Integration and Deployment

All the described components were deployed using Docker containers in the robot laptop (for CartesI/O, AprilTag detection, human tracking), the teleoperation station (for devices management and communication), the Jetson Nano (for streaming webcams), and the inference machine (for microphone management, speech-to-text, LLM-plan, and text-to-speech networks). They communicate using ROS (Noetic).

We relied on the `smach` library to configure and execute robot behaviours as Finite-State Machines (FSMs). In particular, the CartesI/O Python ROS client sends commands to the WBC, and the standard ROS actions or services to send goals to the navigation node, or control the grippers. We employed two basic types of `smach` states for motion: *way-points* (where the indicated end-effector follows a sequence of target points) or *demo-playback* (where the end-effector replicate a demonstrated trajectory recorded via teleoperation, Sec. II-D). In both cases, the user is able to specify the reference frame in which the commanded trajectory is expressed, making it immediate to define motions as offsets from a particular frame of interest (e.g., AprilTag or tracked human poses). Such states are the main building blocks of the a set of sub-FSMs that solve individual portions of the instruction and that can be assembled through specific functions. Specifically, these are: `pick_from_<loc>(object)`, `place_at_<loc>(object)`, `go_to(loc)`, and `handover` (where `loc` is either `table`, `dishwasher`, or `cabinet`, and `object` is a label from the pre-defined set of objects). Different motion specifications (targets, times, demos, etc.) are stored in a configuration file and labelled according to the task nature, following a uniform pattern.

Hence, we employ an "orchestrator" node (running on the CartesI/O container) that requests the plan in JSON format from the LLM (via a ROS service call), and uses the its content to generate programmatically (by concatenating the proper sub-FSMs) the correct FSM and run it to carry out the received instruction. Throughout the *coopetition*, in case the robot was unable to successfully complete the assigned task, teleoperation served as a fallback solution to handle failure cases.

IV. Discussion

Our LLM-based plan generation pipeline robustly understood diverse voice commands from different people with different accents, consistently structuring plans correctly despite phrasing variations. Our teleoperation interface (integrated with the WBC) enabled diverse tasks and high-quality demonstrations but required significant expertise; addressing workspace and kinematic constraints could greatly improve user experience. AprilTag-based perception and navigation were not always robust and required extensive environmental structuring. Extending these approaches to a larger number of locations and objects would be impractical. Therefore, upgrading these modules with state-of-the-art object tracking is essential for future work [12]. *Demo-playback* proved effective for performing complex interactions, such as opening a dishwasher. However, it lacks generalization capabilities, for example, to different dishwasher types. The existing demonstration recording pipeline can be leveraged to collect data-sets for training more robust and more generic policies based on diffusion [29] or flow matching [30] which could handle a greater variety of environments. Finally, we plan upgrading the system to ROS2.

REFERENCES

[1] M. Matamoros, V. Seib, R. Memmesheimer, and D. Paulus, "Robocup@ home: Summarizing achievements in over eleven years of competition," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2018, pp. 186–191.

[2] C. Zhang, J. Chen, J. Li, Y. Peng, and Z. Mao, "Large language models for human–robot interaction: A review," *Biomimetic Intelligence and Robotics*, vol. 3, no. 4, p. 100131, 2023.

[3] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," in *International Conference on Computer Vision (ICCV)*, 2023.

[4] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley, Z. Xu, D. Sadigh, A. Zeng, and A. Majumdar, "Robots that ask for help: Uncertainty alignment for large language model planners," in *Conference on Robot Learning (CoRL)*, 2023.

[5] E. M. Hoffman, S. Caron, F. Ferro, L. Sentis, and N. G. Tsagarakis, "Developing humanoid robots for applications in real-world scenarios [from the guest editors]," *IEEE Robotics & Automation Magazine*, vol. 26, no. 4, pp. 17–19, 2019.

[6] A. Laurenzi, E. M. Hoffman, L. Muratore, and N. G. Tsagarakis, "Cartesi/o: A ros based real-time capable cartesian control framework," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 591–596.

[7] E. M. Hoffman, A. Laurenzi, and N. G. Tsagarakis, "The open stack of tasks library: Opensot: A software dedicated to hierarchical whole-body control of robots subject to constraints," *IEEE Robotics & Automation Magazine (RAM)*, pp. 2–12, 2024.

[8] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," in *Conference on Robot Learning (CoRL)*, 2024.

[9] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, 2019, pp. 614–619.

[10] G. Li, F. Caponetto, E. Del Bianco, V. Katsageorgiou, I. Sarakoglou, and N. G. Tsagarakis, "Incomplete orientation mapping for teleoperation with one dof master-slave asymmetry," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5167–5174, 2020.

[11] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[12] Y. Labbé, L. Manuelli, A. Mousavian, S. Tyree, S. Birchfield, J. Tremblay, J. Carpentier, M. Aubry, D. Fox, and J. Sivic, "Megapose: 6d pose estimation of novel objects via render & compare," in *Conference on Robot Learning (CoRL)*, 2022.

[13] M. Krogius, A. Haggenmiller, and E. Olson, "Flexible layouts for fiducial tags," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[14] F. Amadio, M. Laghi, L. Raiano, F. Rollo, A. Zunino, G. Raiola, and A. Ajoudani, "Target-referred dmps for learning bimanual tasks from shared-autonomy telemanipulation," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 496–503.

[15] A. Farhadi and J. Redmon, "Yolov3: An incremental improvement," in *Computer Vision and Pattern Recognition (CVPR)*, vol. 1804, 2018, pp. 1–6.

[16] Y. Xu, J. Zhang, Q. Zhang, and D. Tao, "ViTPose: Simple vision transformer baselines for human pose estimation," in *Advances in Neural Information Processing Systems*, 2022.

[17] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.

[18] M. Contributors, "Openmmlab pose estimation toolbox and benchmark," https://github.com/open-mmlab/mmpose, 2020.

[19] T. Hempel, A. A. Abdelrahman, and A. Al-Hamadi, "6d rotation representation for unconstrained head pose estimation," in *2022 IEEE International Conference on Image Processing (ICIP)*, 2022, pp. 2496–2500.

[20] S. Macenski, T. Moore, D. V. Lu, A. Merzlyakov, and M. Ferguson, "From the desks of ros maintainers: A survey of modern & capable mobile robotics algorithms in the robot operating system 2," *Robotics and Autonomous Systems*, vol. 168, p. 104493, 2023.

[21] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril *et al.*, "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," *arXiv preprint arXiv:1805.10190*, pp. 12–16, 2018.

[22] J. Wei, X. Wang, D. Schuurmans, M. Bosma, brian ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems*, 2022.

[23] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[24] A. Grattafiori, A. Dubey *et al.*, "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[25] G. Eren and The Coqui TTS Team, "Coqui TTS," Jan. 2021. [Online]. Available: https://github.com/coqui-ai/TTS

[26] T. B. Brown, B. Mann *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, 2020.

[27] D. Totsila, Q. Rouxel, J.-B. Mouret, and S. Ivaldi, "Words2contact: Identifying support contacts from verbal instructions using foundation models," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2024.

[28] B. T. Willard and R. Louf, "Efficient guided generation for llms," *arXiv preprint arXiv:2307.09702*, 2023.

[29] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[30] Q. Rouxel, A. Ferrari, S. Ivaldi, and J.-B. Mouret, "Flow matching imitation learning for multi-support manipulation," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2024, pp. 528–535.